SYSTEM AND METHOD OF PROCESSING DOCUMENTS WITH DOCUMENT PROXIES

This is a continuation, of application Ser. No. 08/210,846, filed Mar. 21, 194, now abandoned.

COPYRIGHT NOTIFICATION

FIELD OF THE INVENTION

The present invention generally relates to computer systems, and more particularly to a method and system for managing document proxies in a document processing system.

BACKGROUND OF THE INVENTION

Object oriented programming (OOP) is the preferred environment for building user-friendly, intelligent computer software. Key elements of OOP are data encapsulation, inheritance and polymorphism. These elements may be used to generate a graphical user interface (GUI), typically characterized by a windowing environment having icons, mouse cursors and menus. While these three key elements are common to OOP languages, most OOP languages implement the three key elements differently.

Examples of OOP languages are Smalltalk, Object Pascal and C++. Smalltalk is actually more than a language; it might more accurately be characterized as a programming environment. Smalltalk was developed in the Learning Research Group at Xerox's Palo Alto Research Center (PARC) in the early 1970s. In Smalltalk, a message is sent to an object to evaluate the object itself. Messages perform a task similar to that of function calls in conventional programming languages. The programmer does not need to be concerned with the type of data; rather, the programmer need only be concerned with creating the right order of a message and using the right message. Object Pascal is the language used for Apple's Macintosh® computers. Apple developed Object Pascal with the collaboration of Niklaus Wirth, the designer of Pascal. C++ was developed by Bjarne Stroustrup at the AT&T Bell Laboratories in 1983 as an extension of C. The key concept of C++ is class, which is a user-defined type. Classes provide object oriented programming features. C++ modules are compatible with C modules and can be linked freely so that existing C libraries may be used with C++ programs. The most widely used object based and object oriented programming languages trace their heritage to Simula developed in the 1960s by O-J. Dahl, B. Myhrhaug and K. Nygard of Norway. Further information on the subject of OOP may be had by reference to Object Oriented Design with Applications by Grady Booch, The Benjimin/Cummings Publishing Co., Inc., Redwood City, Calif. (1991).

With the brief overview of OOP above in mind, document processing has virtually revolutionized the way society generates publications. Typical prior art document processing systems run on top of operating systems, such as DOS. More recently, these document processing systems have been designed to run in a Windows environment. Many of these document processing systems are commercially available. While these document processing systems have improved the ability to process documents and text, there is great inconsistency among document processors with respect to processing methodologies. The result of these inconsistencies creates problems for both application developers and users of the applications.

Application developers must continuously "reinvent the wheel" when creating a new document processor. While operating systems and interface programs provide some tools which may be used, the great majority of the design process for a particular document processor is directed toward creating a group of processing modules which cooperate to allow a user to process documents. Application developers often design processing modules which have already been developed by another company. This requires great duplication of effort, and requires each developer to deal with the details of how to implement various desired functions.

Most graphical computer interface systems provide a user with an interface presented on a graphical display and access to information in one or more graphically presented entities—"documents"—, e.g. a word processor document that allows the user to read and edit the contained textual data. Several graphical computer user interface systems available today, like that of the Apple® Macintosh® computer, provide the user with the capability of graphically managing and organizing multiple document entities represented as small manipulable graphic entities, e.g. "icons". An example of a function in such a system is the ability for the user to request the movement of a document from one containing entity to another by graphically dragging the iconic representation of the document from one document and dropping it onto another document. On currently available systems that support both of the above categories of functions, the system makes the two categories of functions available in disjoint modes of operation. For instance, on the Apple Macintosh, a user must deviate from the mode of editing an open document by switching to the Finder application to perform the document management functions such as moving a document. No system that applicant is aware of has a proxy function for integrating document processing into basic system operations.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide a document processing system utilizing a unique feature terms a "proxy". A proxy integrates external document management functions simultaneously and seamlessly into the standard operating system document processing commands. This system and method provides an interface supporting document access and editing functions from within a document or other active application.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a personal computer system in accordance with a preferred embodiment of the invention;

FIG. **2** is an illustration of a display with a document proxy in accordance with a preferred embodiment of the invention;

FIG. **3** is a data structure of information associated with a typical document stored in accordance with a preferred embodiment;

FIG. **4** is a schematic that depicts a desktop screen in accordance with a preferred embodiment;

FIG. **5** is a schematic with drop-accepting objects or target regions highlighted in accordance with a preferred embodiment;